

/\*

Ant Project. Otago Polytechnic, New Zealand. 2009 3rd Year B.I.T. project for  
Jun Cui, Gareth Dorset and Trevor Farquharson  
-::~\*.\_.\*~:-

Client: Otago Museum

Developers: Jun Cui - 3rd year student Otago Polytechnic  
Gareth Dorset - 3rd year student Otago Polytechnic  
Trevor Farquharson - 3rd year student Otago Polytechnic

Mentors: Patricia Haden - Otago Polytechnic  
Hamish Smith - Otago Polytechnic  
Sam Mann - Otago Polytechnic

The TAnt class is based on the Finite State Machine to control the ant behaviour.  
The Transition method dictates when to change state  
The Action method performs the ant states behaviour.  
The Movement method moves the ants around the ant area

The ants have 8 possible states.

1. searching // searching for food, people or pheromone trails.
2. interacting // interacting with people via the motion detection part of the program
3. goingToFood // going to a food item that is within their visrange
4. pickingUpFood // picking up some of the food they have found and are at.
5. goingHome // returning to the nest either with or without food.
6. inHome // when they are in the nest
7. followingTrail // following a pheromone trail they have found towards a food item
8. leavingHome // this has the ants heading away from the nest before  
// changing to another state but still check for pheromone trails.

\*/

```
package {
    // Imports
    import flash.display.*;
    import flash.events.*;
    import flash.geom.*;

    // Ant Class
    public class TAnt extends MovieClip {
        // Constants
        // States
        private const _search:          Number = 0;          // default state, ant is wandering
        around looking                  // for people, pheromone trail or food
        private const _interact:        Number = 1;          // ant has found a person and is
        interacting                      // with them (move towards)
        private const _goToFood:        Number = 2;          // ant has seen food and is going
        to it                           // ant is at food and is picking
        private const _pickUpFood:      Number = 3;
        some up                         // ant is heading home, could be
        private const _goHome:          Number = 4;
        holding food,                   // because it has been out too long
        private const _inHome:          Number = 5;          // ant is in home (initial state)
        private const _followTrail:     Number = 6;          // ant has found a pheromone trail
        and                             // is following it to food
        private const _leaveHome:       Number = 7;          // ant has just exited home and is
        moving                          // away before searching

        // Attributes
        private const _visRange:        Number = 100;        // visual range of ants, determines how
        close to                         // food before finding it
        private const _sceRange:        Number = 50;         // scent range, determines how close to
        pheromone                       // trail before finding it
        private const _persRange:       Number = 300;        // person detection range, how close the
        ant has                          // to be to the person before interacting
        private const _orientProb:      Number = 25;         // probability of orienting to a target
        private const _speed:           Number = 2;          // movement speed of ants
        private const _homeX:           Number = 640;        // ant knows where it's home is
        private const _homeY:           Number = 44;         // see above comment.

        // Variables
        private var antState:           Number;              // denotes which state the ant is
        currently in                    // if ant is heading
        private var targetFood:         TFood;
        towards food, this signifys the // target food item
        private var cycleCount:         Number;              // general counter used
        for changnig some states
        private var holdingFood:        Boolean;             // denotes whether the ant has
        picked up food
        private var pathCreator:        Boolean;             // denotes whether the ant is the
        first to reach a food item
        private var screenWidth:        Number;              // width of the ant area, used for
        bounds checking
        private var screenHeight:       Number;              // height of the ant area, also
        used for bounds checking
    }
}
```

```

        private var lkX: Number; // last known position of
the person, used to stagger loss of

        private var lkY: Number; // interest in person // see above comment.

// Constructor
public function TAnt(sW: Number, sH: Number, rCyc: Number) {
/*
    Method: TAnt -> Constructor
    Description: creates and initialises the ant variables
*/
    cacheAsBitmap = true; // optimisation so the
ants move smoother
    rotation = Math.random() * 360; // set ants to point in a
random direction
    pathCreator = false; // this is set to true when an ant first
reaches a food item
    holdingFood = false; // set to true when an ant picks up food.
    targetFood = null; // Ant property. food item that an ant has
found or is // following a trail to.
    antState = _inHome; // start the ants
in their nest
    SetScale(0); // in nest, so not
visible (scale = 0)
    x = _homeX; //
    y = _homeY; // initialise y. location to nest
    screenWidth = sW; // width of the ant area, used for bounds
checking
    screenHeight = sH; // height of the ant area, used for bounds
checking
    cycleCount = rCyc; // counter for keeping track of frame
count // before changing state
}

// Set and Get Methods
public function SetScale(nScale: Number): void {
/*
    Method: SetScale
    Description: alters the size of the ant, common values to
    modify to are 0 (invisible) and 1 (fullsize)
*/
    scaleX = nScale;
    scaleY = nScale;
}

public function GetlkX(): Number {
/*
    Method: GetlkX
    Description: returns the last known x position of the person
*/
    return lkX;
}

public function GetlkY(): Number {
/*
    Method: GetlkY
    Description: returns the last known y position of the person
*/
    return lkY;
}

public function SetCycles(nCycles: Number): void {
/*
    Method: SetCycles
    Description: sets the cycle count.
    uses random number so each ant will have a slightly
    different cyclecount from the other ants
*/
    var mul = nCycles / 5;
    cycleCount = Math.floor(Math.random() * mul + nCycles);
}

// Utilities
public function DistToObject(object: MovieClip): Number {
/*
    Method: DistToObject
    Description: returns the direct distance to a target
    object from an ant.
    This uses Pythagoras' Theorem.
*/
    // get x and y difference between an ant and an object.
    // object can be food, another ant.
    var dX = x - object.x;
    var dY = y - object.y;

    // then get the squared value of both the distances so
    // it can be used to get the square root
    var sqrd = (dX * dX) + (dY * dY);

    // now we get the square root and then round down
    // to the nearest whole number
    var distOb = Math.floor(Math.sqrt(sqrd));
    // and return that distance.

```

```

        return distOb;
    }

    public function DistToTarget(tX: Number, tY: Number): Number {
        /*
            Method: DistToTarget
            Description: returns the distance to a target location
        */
        // gets the x and y difference between an ant and an
        // inputted x and y location.
        // the x and y location are passed in.
        var dX = x - tX;
        var dY = y - tY;
        // then get the squared value of both the distances so it can be
        // used to get the square root
        var sqrd = (dX * dX) + (dY * dY);
        // now we get the square root and then round down to the
        // nearest whole number
        var distTa = Math.floor(Math.sqrt(sqrd));
        // and return that distance
        return distTa;
    }

    // this checks that an ant is further than 60 pixels away from the home
    // or any corner of the ant area. If they are further than 60 pixels away
    // then the ant Collision method is called in the Transition method.
    // This helps prevent the ants getting stuck in confined / busy areas.
    public function ACCheck(): Boolean {
        // assume that we are not going to call the Collision method
        var retVal = false;
        // if they are not near the ant nest/home
        if ((DistToTarget(_homeX, _homeY) > 60)
            // and the top left corner
            && (DistToTarget(0, 0) > 60)
            // and the top right corner
            && (DistToTarget(screenWidth, 0) > 60)
            // and the bottom left corner
            && (DistToTarget(0, screenHeight) > 60)
            // and the bottom right corner
            && (DistToTarget(screenWidth, screenHeight) > 60))
        {
            // if we are not near any of the corners or nest then set to true.
            retVal = true;
        }

        return retVal; // return this boolean value
    }

    public function AntCollision(ants: Array): void {
        /*
            Method: AntCollision
            Description: checks the distance to the other ants, and alters the direction
            to avoid running over each other. For each of the ants, it gets the distance,
            and the angle to the other ant and if it is determined the ants are
            going to collide, it adjusts the direction of the ant calling this function
            based on what the angle to the other ant is.
        */
        for (var i in ants) {
            // get distance to the other ant.
            var distAnt = DistToObject(ants[i]);
            // ant distance of 0 indicates the ant is checking itself, so don't take any
            // ** Important -> ants run in circles otherwise **
            if (distAnt != 0) {
                // get angle to other ant in Radians
                var antRad = Math.atan2(ants[i].y - y, ants[i].x - x);
                // then convert to degrees
                var antDeg = 180 + ((Math.round(rotation + 90) - Math.round((antRad /
                    0.01745))) % 180);

                // if the ant is close to the other ant, then we have to rotate it's
                // away from the other ant
                if (distAnt < _sceRange * 1.5) {
                    // if the other ant is within 80 degrees to the LEFT of the ant
                    // in relation to it's own current direction, being 0 degrees -
                    // then adjust the ant checking direction.
                    if ((antDeg > -80) && (antDeg < 0)) {
                        // add 90 to the ant angle and then divide by 3 so the
                        // change is not dramatic.
                        rotation += (90 + antDeg) / 3;
                    }

                    // if the other ant is within 80 degrees to the RIGHT of the ant
                    // in relation to it's own current direction, being 0 degrees -
                    // then adjust the ant checking direction.
                    if ((antDeg > 0) && (antDeg < 80)) {
                        // add -90 to the ant angle and then divide by 3 so the
                        // change is not dramatic.
                        rotation += (-90 + antDeg) / 3;
                    }
                }
            }
        }
    }

```

```

their

// if the ants are heading directly towards each other, change
// course so they aren't
if(antDeg == 0) {
    // rotaion is the current angle 2 degrees to the left.
    rotation -= 2;
}
}
}

}

// Transition Method is the main controlling function of the ants.
// It checks for all the conditions required to change the behaviour of the ants.
public function Transition(ants: Array, food: Array, pX: Number, pY: Number): void {
/*
    Method: Transition
    Description: checks for conditions to alter the state the ant is in.
*/
    // Local variables
    // distance to food variable
    var dFood: Number

    // adjusts the movieClip frames if the ant is holding food
    if(holdingFood) {
        // frames 5 - 8 show ant holding food
        if(currentFrame < 5) {
            // play from frame 5
            gotoAndPlay(5);
        }
    }

    // adjusts the movieClip frames if the ant is not holding food.
    } else {
        // frames 1 - 4 show ant normally, ie. no food
        if(currentFrame > 4) {
            // play from frame 1
            gotoAndPlay(1);
        }
    }

    // depending on what state the ant is in, determines what state the ant
    // can check and transition to.
    switch (antState) {
        // searching for food, pheromone trails or a person.
        case _search :
        {
            if(ACCheck()) {
                // don't check for collisions if near home or corners of screen
                // as lots of ants are likely to get stuck here.
                AntCollision(ants);
            }
            // if the ant has been out searching for a set time and has
            // not gone into any other state
            if (cycleCount == 0) {
                // then tell the ant to goHome
                antState = _goHome;
            } else {
                // decrease the cycle count if the ant is still out.
                cycleCount--;
            }
            // call method to check if food is within visual range
            FoodVisCheck(food);
            // call method to check if pheromone trail is within scent range.
            FoodSceCheck(food);

            // check if a person, via motion detection is near by.
            CheckPerson(pX, pY);
            break;
        };

        // interacting with a person
        case _interact :
        {
            // get the distance between the ant and the person
            var distPerson = DistToTarget(pX, pY);
            // and if the person is still within the ant area, update the persons
            // last known x and y position.
            if (pX > 0) {
                // pX shoots off to -500 after 10 seconds of inactivity,
                // so while it's greater than 0, set lkX and lkY
                lkX = pX;
                lkY = pY;
            }
            // if the person is with outside the ants persRange
            if (distPerson > _persRange) {
                // decrease the cycleCount
                cycleCount--;
                // and if the cycle count is 0 or the person is out if the
                // persRange but still in the ant area
                if ((cycleCount == 0) || (pX > 0)) {
                    // change the ant state to search.
                    antState = _search;
                    // reset the cycle count
                    SetCycles(5000);
                }
            }
        }
    }
}

```

```

        break;
    };

    // if ant is going to food. The ant is with sight
    // - visRange - of the food
    case _goToFood :
    {
        // if the food is all gone - the foods movieClip frame number
        // is at or above 42
        if (targetFood.GetCurrFrame() >= 42) {
            // then make sure the frame is at 42
            targetFood.ChangeFrame(42);
            // change the ant state to search
            antState = _search;
            // reset the cycle count to 5000
            SetCycles(5000);
        }

        // get the distance to the food
        dFood = DistToObject(targetFood);
        // if the ant is within 3 pixels and the food movieClip
        // frame number is less than 42
        if ((dFood <= 3) && (targetFood.GetCurrFrame() < 42)) {
            // change ant state to pickUpFood, so it picks up the food
            antState = _pickUpFood;
            // set cycle count to 10, this determines how long the
            // ant is in the pickUpFood state
            SetCycles(10);
            // get the frame that the food is currently on
            var foodFrame = targetFood.GetCurrFrame();
            // and increment the food frame by 1 and assign to the
            // target food item
            targetFood.ChangeFrame(foodFrame + 1);
        }
        break; // end goToFood state
    };

    // if the ant is at the food and is picking some up
    case _pickUpFood :
    {
        // if the food frame is gone and we the ant hasn't managed
        // to pick some food up
        if ((targetFood.GetCurrFrame() > 42)&&(!holdingFood)) {
            // then make the food movieClip frame is at 42
            targetFood.ChangeFrame(42);
            // set the ant to search
            antState = _search;
            // set cycle count to 5000
            SetCycles(5000);
        } else {
            // otherwise the food is still there
            // set holdingFood to true
            holdingFood = true;
            // check if the food has a pheromone trail associated with it
            if (!targetFood.GetHasPath()) {
                // and if not, then make this ant the pheromone trail
                pathCreator = true;
            }

            //if the cycle count is greater than 0
            if (cycleCount != 0) {
                // decrement cycleCount
                cycleCount--;
            } else {
                // otherwise tell the ant to go home
                // go home ant!!
                antState = _goHome;
                // and set cycle count which is used in the Action
                // the ant when to update the pheromone trail path
                SetCycles(10);
            }
        }
        break; // end pickUpFood
    };

    // if the ant is going home
    case _goHome :
    {
        // check the ant is not going to run over other ants
        if(ACCheck()) {
            AntCollision(ants);
        }

        // if the ant is within 10 pixels of the home location
        if (DistToTarget(_homeX, _homeY) <= 10) {
            // change state to inHome
            antState = _inHome;
            // set the ant scale size to 0 so it is invisible
            SetScale(0);
            // tell the ant it is no longer holding food.
            // Play frames 1 to 4 in the ant movieClip
            holdingFood = false;
            // set cycles
            SetCycles(120);
        }
    }
}

```

pathCreator

method to tell

```

        }
        // if the ant is not holding food, check if food is with
        // the ants visual range
        if (!holdingFood) {
            FoodVisCheck(food);
        }
        break; // end goHome
    };

    // if the ant is inHome
    case _inHome :
    {
        // if the cycleCount is greater than 0
        if (cycleCount != 0) {
            // decrement it
            cycleCount--;

            // otherwise the cycleCount is 0
        } else {
            // set the ant so it is not a pathCreator
            pathCreator = false;
            // set targetFood to null so it cannot try to go to any food
            targetFood = null;
            // set state to leave home
            antState = _leaveHome;
            // point away from our direction coming in for when we leave
            rotation += 180;
            // make the ant visible
            SetScale(1);
            // this makes the ant walk straight out the ant nest without any
            // collision checking so the ants don't get stuck in a busy area
            SetCycles(45);
        }
        break; // end inHome
    };

    // if the ant is following a pheromone trail towards the food
    case _followTrail :
    {
        // checking if there is any closer food along the pheromone trail
        FoodVisCheck(food);
        // check the ant is not going to run over other ants
        if (ACCheck()) {
            AntCollision(ants);
        }

        // get the distance to the food
        dFood = DistToObject(targetFood);
        // if the food is with visual range, can see the food, so go to it
        if (dFood < _visRange) {
            // set ant state to goToFood
            antState = _goToFood;
        }
        // get the strength of the pheromone trail
        var trailStr = targetFood.GetPathStr();
        // if the strength is less than 5, indicates the trail is
        // too weak to follow
        if (trailStr < 5) {
            // set the ant to search
            antState = _search;
            // set cycle count
            SetCycles(5000);
        }
        // and check to see if the ant is near a person
        CheckPerson(pX, pY);
        break; // end followTrail
    }

    // if the ant is leaving the nest
    case _leaveHome :
    {
        // check if a pheromone trail is near by
        FoodSceCheck(food);
        // if the cycle count is 0
        if (cycleCount == 0) {
            // then set the ant state to search
            antState = _search;
            // and reset cycle count
            SetCycles(5000);
        }

        // otherwise decrement the cycle count
        else {
            cycleCount--;
        }
        break; // end leaveHome
    };
} // end switch statement

// if the person is still within the area, call the Action method.
if (pX > 0) {
    // call action method with the person's x and y,
    Action(pX, pY);
} else {
    // otherwise, if person is out of area, call action with last known x and y
    Action(lkX, lkY);
}

```

```

}
} // End Transition

public function CheckPerson(pX: Number, pY: Number) : void {
/*
    Method: CheckPerson
    Description: called from transition method, this checks whether a person is
    within range of the ant and changes state if necessary
*/
    // if the person is with the ant area
    if((pX > 0) || (pX < screenWidth)) {
        // get the distance to the person
        var distPerson = DistToTarget(pX, pY);
        // and if the person is within range of the ant
        if (distPerson < _persRange) {
            cycleCount = Math.floor(Math.random() * 250);
            // set the ant state to interact
            antState = _interact;
        }
    }
}

// check if a food item is within visual range
public function FoodVisCheck(food: Array): void {
/*
    Method: FoodVisCheck
    Description: called from transition method, this checks whether a food item
    is within visual range of the ant and changes state if necessary
*/
    // iterate thru the food array
    for (var j in food) {
        // get the distance to the food
        var distFood = DistToObject(food[j]);
        // and if the food is within visual range and there is still food to pickup
        if ((distFood < _visRange) && (food[j].GetCurrFrame() <= 41)) {
            // set the ants targetFood to be that food item
            targetFood = food[j];
            // and tell the ant to go toward that food item.
            antState = _goToFood;
        }
    }
}

// check if a pheromone trail is within scent range
public function FoodSceCheck(food: Array): void {
/*
    Method: FoodSceCheck
    Description: called from transition method, this checks whether a pheromone
    trail is within range of the ant and changes state if necessary
*/
    // iterate thru the food array
    for (var j in food) {
        // check if a food item hasPath boolean is set to true
        if (food[j].GetHasPath()) {
            // get the x distance from the food to the end of the pheromone trail
            var dX = food[j].GetX() - food[j].GetPathX();
            // get the y distance from the food to the end of the pheromone trail
            var dY = food[j].GetY() - food[j].GetPathY();
            // then calculate the angle of the trail
            var pathAngle = Math.atan2(dY, dX);
            // if the ant is between the food and the end of the trail in the y
            direction
            if ((this.y < food[j].GetY()) && ((this.y > food[j].GetPathY()))) {
                // then get the ant x position
                var xIntercept = this.x;
                direction
                // and get the distance the ant is from the food in the x
                var dX2 = food[j].GetX() - xIntercept;
                direction
                // and get the distance the ant is from the trail in the y
                var dY2 = dX2 * Math.tan(pathAngle);
                // and get the point on the trail intersecting with the ant y
                location
                var yIntercept = food[j].GetY() - dY2;
                // then get the distance the ant is from the nearest point of
                the path
                var distPath = DistToTarget(xIntercept, yIntercept);
                // if the path is within the ants scentRange
                if (distPath < _sceRange) {
                    // set the food item to be the ants targetFood
                    targetFood = food[j];
                    // and set the ant state to followTrail
                    antState = _followTrail;
                }
            }
        }
    }
}

// Action Method called from the Transition method
public function Action(pX: Number, pY: Number): void {
/*
    Method: Action
    Description: this performs the action required based on what
    state the ant is in
*/

```

```

switch (antState) {
    // if the ant is searching then
    case _search :
    {
        // call this method which randomly alters the direction the ant
        // is facing and moves the ant
        ChangeDir();
        break;
    };
    // if the ant is interacting
    case _interact :
    {
        // orientate the ant to the persons x and y position
        OrientTo(pX, pY);
        break;
    };
    // if the ant is going to food
    case _goToFood :
    {
        // orientate the ant to the targetFood x and y position
        OrientTo(targetFood.x, targetFood.y);
        break;
    };
    /*case _pickUpFood:
    {
        no current actions for pickUpFood state
        break;
    };*/
    case _goHome : // if the ant is going home
    {
        // check if the ant is holding food
        if (holdingFood) {
            // check if they are the pathCreator
            if (pathCreator) {
                // update where the end of the pheromone trail is when
                if (cycleCount == 0) {
                    // set the x location of the trail
                    targetFood.SetPathX(x);
                    // set the y location of the trail
                    targetFood.SetPathY(y);

                    // set the strength of the trail.
                    targetFood.SetPathStr(600);
                    // reset cycleCount, pathcreator ant will update

                    // position every 10 frames
                    cycleCount = 10;

                } else {
                    // otherwise decrement the cycleCount, meaning
                    // end position is not set to be updated yet.
                    cycleCount--;
                }
            }
            // otherwise the ant is not the pathCreator
        } else {
            // and they just increase the strength of the trail.
            var str = targetFood.GetPathStr() + 1;
            // set the trails new strength
            targetFood.SetPathStr(str);
        }
    }
    // orientate the ant to home
    OrientTo(_homeX, _homeY);
    break;
};
/*case _inHome:
{
    No current actions for inHome state
    break;
};*/
// if ant is following the trail
case _followTrail :
{
    // orientate the ant to the targetFood
    OrientTo(targetFood.x, targetFood.y);
    break;
};
// if ant is leaving home
case _leaveHome :
{
    // move the ant in the direction they are facing.
    Move();
    break;
};
};
}

// Movement Methods
// change ant direction
public function ChangeDir(): void {
    /*
        Method: ChangeDir
        Description: this alters the direction the ant is facing at random intervals
    */

```



```

        // the ant has a certain percentage chance of changing direction
        // and also a 50/50 chance of turning left or right
        // if the random number is less than the percentage of change
        if (Math.random() < 0.2) {
            // rotate the ant a random number of degrees
            var rotate = 5 + (Math.random() * 50);
            // if random number is less than 0.5
            if (Math.random() < 0.5) {
                // then turn the ant left
                rotate *= -1;
            }
            // apply the rotation to the ant
            this.rotation += rotate;
        }
        // move the ant
        Move();
    }

    // gets the angle to a target point and makes the ants face towards the target
    public function OrientTo(xDest: Number, yDest: Number):void {
        /*
            Method: OrientTo
            Description: this orients the ant towards a target destination
        */
        var currRad: Number;
        var currDeg: Number;
        // random probability of pointing to the object
        var pr = _orientProb / 100;
        // if the random number is less than the probability
        if (Math.random() < pr) {
            // get the angle in Radians to the target
            currRad = Math.atan2(yDest - y, xDest - x);
            // convert it to degrees
            currDeg = Math.round(currRad * 180 / Math.PI);
            // set the ants rotation ( + 90 required as the movieClip
            // rotation is 90 degrees out from the stage rotation
            this.rotation = currDeg + 90;
        }
        // adds a drunken walk of upto 5 degrees to make the ant walk look more real
        var rotate = Math.random() * 5;
        // if random number is less than 0.5
        if (Math.random() < 0.5) {
            // turn the ant left
            rotate *= -1;
        }
        // apply the rotation
        this.rotation += rotate;
        // move the ant
        Move();
    }

    // moves the ant
    public function Move() {
        /*
            Method: Move
            Description: this moves the ant in the direction it's facing
        */
        // convert ants direction to Radians allowing for the 90 degree variance
        // between the movieClip and stage
        var rad = (rotation - 90) * 0.01745;
        // calculate how far in the x direction to move
        var deltaX = _speed * Math.cos(rad);
        // calculate how far in the y direction to move
        var deltaY = _speed * Math.sin(rad);
        // apply the x movement
        x += deltaX;
        // apply the y movement
        y += deltaY;

        // this is for bounds checking so the ants stay in the ant area
        // if the ant is going off the top or bottom of the ant area
        if((y < 0) || (y > screenHeight)) {
            // orientate the ant to it's x position and the middle of the
            // area in the y direction
            OrientTo(x, screenHeight / 2);
        }
        // if the ant is going off the left or right side of the ant area
        if((x < 0) || (x > screenWidth)) {
            // orientate the ant to it's y position and the middle of the
            // area in the x direction
            OrientTo(screenWidth / 2, y);
        }
    } // end Move
} // End Class
} // End Package

```